

CS 161

Design and Analysis of Algorithms

Lecture 1:

Logistics, introduction, and multiplication!

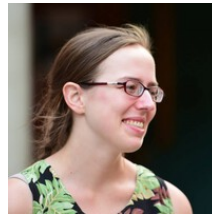
How was your break?

The big questions

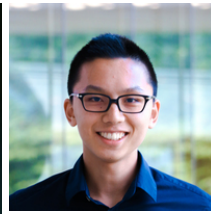
- Who are we?
 - Professor, TAs, students?
- Why are we here?
 - Why learn about algorithms?
- What is going on?
 - What is this course about?
 - Logistics?
- Can we multiply integers?
 - And can we do it quickly?



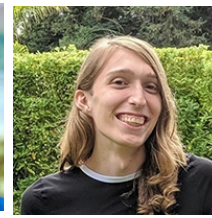
Who are we?



Reyna



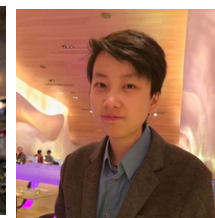
Albert



Amy



Avery



Caci

- Instructors:

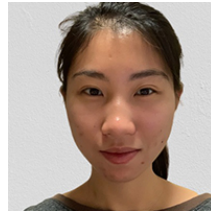
- Moses Charikar
- Nima Anari

- Course Coordinator:

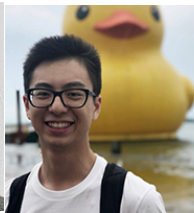
- Amelie Byun



Amelie



Carrie



Changyu



Chenru



Geng



Ian



Jerry

Awesome CAs:

- Reyna Hulett (Head CA)
- Albert Zuo
- Amy Kanne
- Avery Wang
- Caci Jiang
- Carrie Wu
- Changyu Bi
- Chenru Liu
- Geng Zao
- Ian Tullis
- Jerry Qu

- Jesus Cervantes
- Jiazheng Zhao
- John Sun
- Nick Lai
- Ofir Geri
- Qile (Suyie) Zhi
- Rose Li
- Teresa Noyola
- Trey Connelly
- Weiyun (Anna) Ma
- Wilhelm Kautz



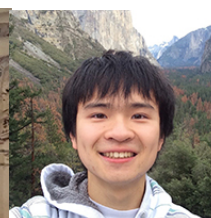
Jesus



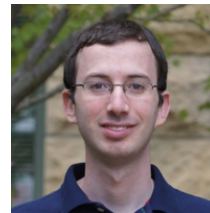
Jiazeng



John



Nick



Ofir



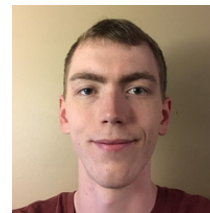
Suyie



Rose



Teresa



Trey



Anna



Wilhelm

Who are you?

- Freshman
- Juniors
- Sophomores
- Seniors
- MA/MS Students
- NDO Students
- PhD Students

Concentrating in:

- Aero/Astro
- Applied Physics
- Art Practice
- Biomedical Informatics
- Chemical Eng.
- Chemistry
- Civil & Env. Eng.
- Classics
- CME
- Comparative Literature
- Computer Science
- Creative Writing
- Digital Humanities
- East Asian Studies
- Economics
- Education
- EE
- Energy Resources Eng.
- English
- Epidemiology
- Ethics in Society
- Fem, Gen, Sex Studies
- French
- Geological Sciences
- Global Studies
- History
- Human Biology
- Human Rights
- Immunology
- Linguistics
- Math
- Modern Languages
- Music
- MS&E
- Mech. Eng.
- Physics
- Philosophy
- Political Science
- Psychology
- Science, Tech. and Society
- Statistics
- Spanish
- Symbolic Systems
- Undeclared

Where are you?

Why are we here?

- I'm here because I'm super excited about algorithms!

Yay Algorithms!

You are better equipped to answer this question than I am, but I'll give it a go anyway...

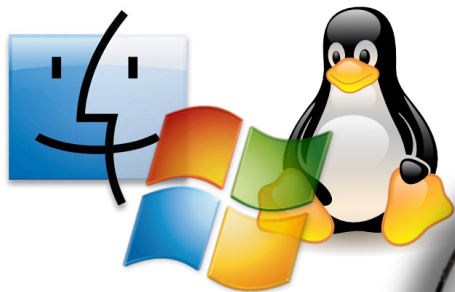
Why are you here?

- Algorithms are **fundamental**.
- Algorithms are **useful**.
- Algorithms are **fun!**
- CS161 is a **required course**.

Why is CS161 required?

- Algorithms are **fundamental**.
- Algorithms are **useful**.
- Algorithms are **fun!**

Algorithms are fundamental



Operating Systems (CS 140)

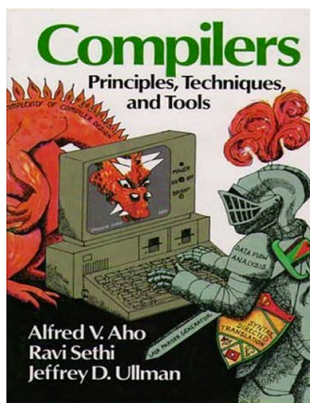


The
Algorithmic
Lens

229)



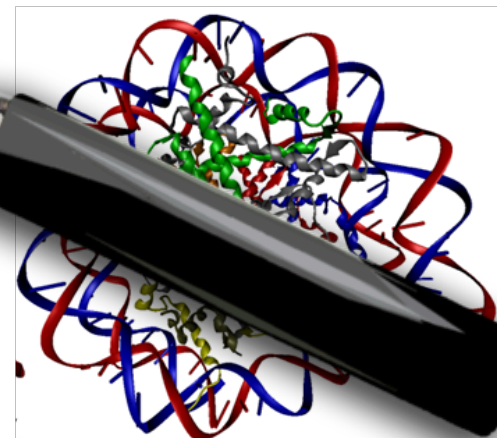
Cryptography (CS 255)



Compilers (CS 143)



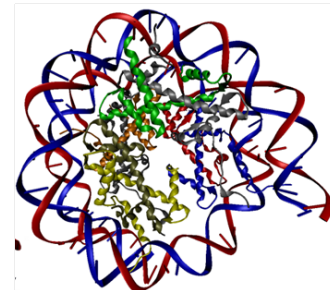
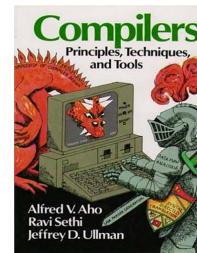
Networking (CS 144)



Computational Biology (CS 262)

Algorithms are useful

- All those things without the course numbers.
- As inputs get bigger and bigger, having good algorithms becomes more and more important!



Algorithms are fun!

- Algorithm design is both an **art** and a **science**.
- Many **surprises!**
- Many **exciting research questions!**

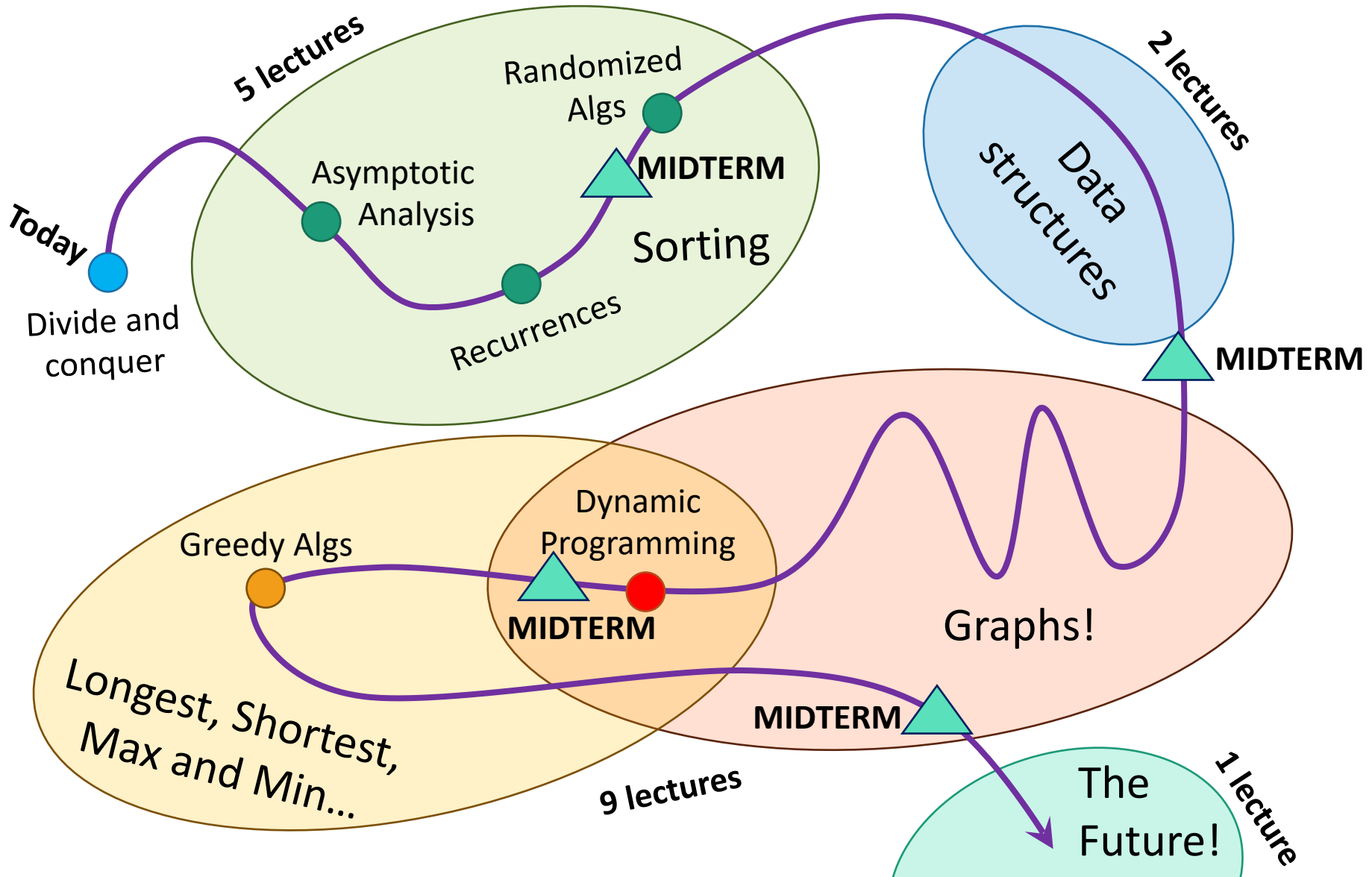
What's going on?

- Course goals/overview
- Logistics

Course goals

- The **design and analysis** of algorithms
 - These go hand-in-hand
- In this course you will:
 - Learn to **think analytically** about algorithms
 - Flesh out an “**algorithmic toolkit**”
 - Learn to **communicate clearly** about algorithms

Roadmap

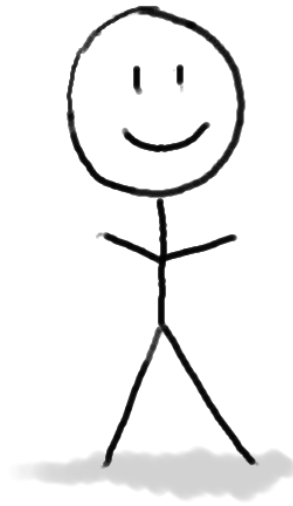


Our guiding questions:

Does it work?

Is it fast?

Can I do better?



Our internal monologue...

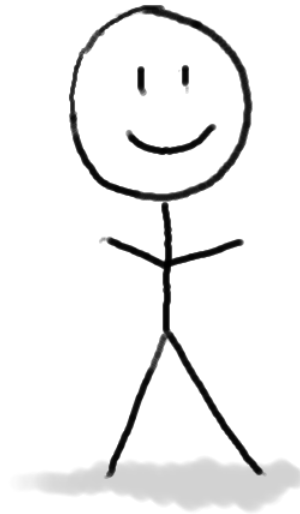
What exactly do we mean by better? And what about that corner case? Shouldn't we be zero-indexing?



Plucky the
Pedantic Penguin

Detail-oriented
Precise
Rigorous

Does it work?
Is it fast?
Can I do better?



Both sides are necessary!

Dude, this is just like that other time. If you do the thing and the stuff like you did then, it'll totally work real fast!



Lucky the
Lackadaisical Lemur

Big-picture
Intuitive
Hand-wavy

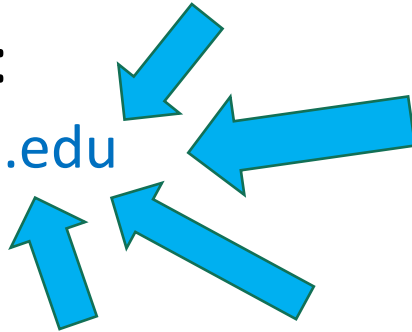
Aside: the bigger picture

- Does it work?
- Is it fast?
- Can I do better?
- Should it work?
- Should it be fast?
- We want to reduce crime.
- It would be more “efficient” to put cameras in everyone’s homes/cars/etc.
- We want advertisements to reach to the people to whom they are most relevant.
- It would be more “efficient” to make everyone’s private data public.
- We want to design algorithms, that work well, on average, in the population.
- It would be more “efficient” to focus on the majority population.

Course elements and resources

- Course website:

- cs161.stanford.edu



- Lectures
- References
- IPython Notebooks
- Homework
- Exams
- Office hours, recitation sections, and Ed

Lectures

- On Zoom (link on canvas), M/W, 10:00-11:20
- Resources available:
 - Slides, Videos, Book, IPython notebooks

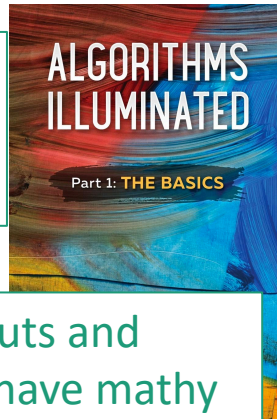
CS 161
Design and Analysis of Algorithms
Lecture 1:
Logistics, introduction, and multiplication!

Slides are the slides from lecture.



Videos from lecture are available!

Hand-outs and references have mathy details that slides may omit



Jupyter Karatsuba Last Checkpoint: 16 hours ago (unaved changes)

File Edit View Insert Cell Kernel Help

Lecture 1: Multiplication

In this IPython notebook, we implement the algorithms that we discussed in class for multiplying integers.

The goal:
Multiply two n-digit integers.

The rules:
We are allowed to use Python's built-in multiplication to do one-digit multiplications (e.g. 6 times 7), but not any other multiplication. We are also allowed to use Python's built-in addition (even of large numbers).

```
In [1]: # First include a few helper functions that will be useful (like turning integers to digits)
from multipliers import *
```

```
In [2]: # this is probably (equivalent to) the algorithm you learned in grade school.
# look at each pair of digits, and add them up with appropriate shifts.
def gradeSchoolMult(x, y): # x and y are integers
    ret = 0
```

IPython notebooks have implementation details that slides may omit.

How to get the most out of lectures

- **During lecture:**

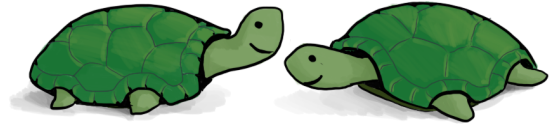
- Participate live (if you can), ask questions.
- Engage with in-class questions.

- **Before lecture:**

- Do **pre-lecture exercises** on the website.

- **After lecture:**

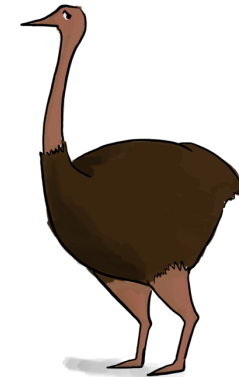
- Go through the exercises on the slides.



Think-~~Pair~~-Share Terrapins
(in-class questions)



Siggi the Studious Stork
(recommended exercises)

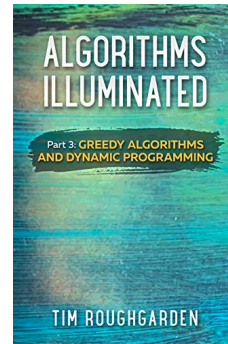
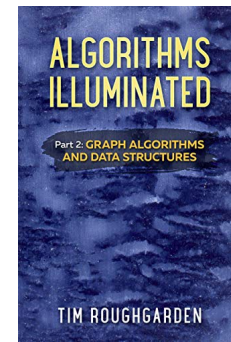
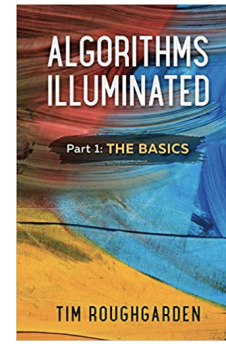


Ollie the Over-achieving Ostrich
(challenge questions)

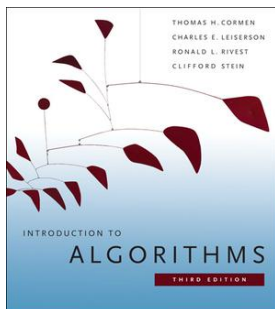
- **Do the reading**

- either before or after lecture, whatever works best for you.
- **do not wait to “catch up” the week before the exam.**

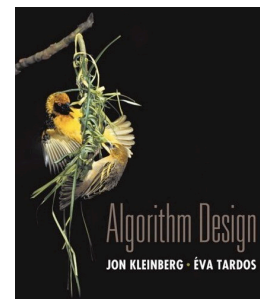
Optional References



- **Algorithms Illuminated**, Vols 1,2 and 3 by Tim Roughgarden
- Additional resources at algorithmsilluminated.org
- We may also refer to to the following (optional) books:



“CLRS”: Introduction to Algorithms by Cormen, Leiserson, Rivest, and Stein. Available FOR FREE ONLINE through the Stanford library.



“Algorithm Design” by Kleinberg and Tardos

IPython Notebooks

- Lectures will occasionally use IPython notebooks (but not homeworks)
 - For next lecture, the *pre-lecture exercise* is to get started with Jupyter Notebooks and with Python.
 - See course website for details.
- The goal is to make the algorithms (and their runtimes) more tangible.

Homework!

- Weekly assignments, posted Wednesday by 12:30pm, due the next Wednesday 11:59pm.
- First HW posted this Wednesday!

How to get the most out of homework

- HW has two parts: exercises and problems.
- Do the exercises on your own.
- Try the problems on your own **before** discussing it with classmates.
- If you get help from a CA at office hours:
 - **Try the problem first.**
 - **Ask: “I was trying this approach and I got stuck here.”**
 - **After you’ve figured it out, write up your solution from scratch, without the notes you took during office hours.**

Exams

- There will be 4 **midterms**.
(2 hour exams to be taken in a 48 hour window)
 - **Midterm 1:** Thu Jan 28 – Fri Jan 29
 - **Midterm 2:** Thu Feb 11 – Fri Feb 12
 - **Midterm 3:** Mon Mar 1 – Tue Mar 2
 - **Midterm 4:** Mon Mar 15 – Tue Mar 16
- We will drop the lowest score of first 3 midterms; last midterm cannot be dropped.
- Weighting: **Homeworks (55%), Midterms (45%)**
- **If you have a conflict with the midterm times, email cs161-win2021-staff@lists.stanford.edu ASAP!!!!**

Talk to us!

ed CS 161 – Discussion

- Sign up for Ed:
 - See [course website \(Resources\)](#) for link: "sign up for Ed"
 - Course announcements will be posted there
 - Discuss material with TAs and your classmates
- Office hours (on Nooks):
 - See course website for schedule
- Recitation sections (on Nooks):
 - Wednesdays?, Thursdays and Fridays?
 - See course website for schedule (to be posted soon)
 - Technically optional, but ***highly recommended!***
 - Extra practice with the material, example problems, etc.

Talk to each other!

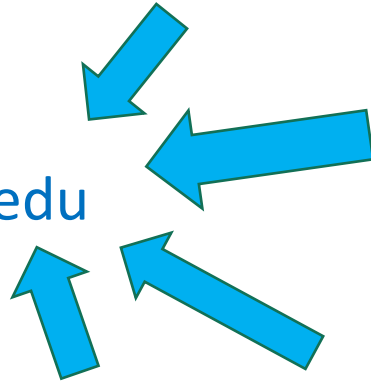
ed CS 161 – Discussion

- Answer your peers' questions on Ed!
- We will host Homework Parties (on Nooks).

Course elements and resources

- Course website:

- cs161.stanford.edu



- Lectures
- References
- IPython Notebooks
- Homework
- Exams
- Office hours, recitation sections, and Ed



A note on course policies

- Course policies are listed on the website.
- Read them and adhere to them.
- That's all I'm going to say about course policies (except for a couple of slides on collaboration and the honor code)

RULES

Collaboration

- We encourage collaboration on homeworks (but strongly recommend you do exercises on your own)
- Valid and invalid modes of collaboration detailed on the course website.
 - Briefly, you can exchange ideas with classmates, but must write up solutions on your own.
- You must cite all collaborators, as well as all sources used (outside of course materials).

Honor code

- Has been updated for AY 2020-21: **“In all cases, it is not permissible for students to enter exam questions into any software, apps, or websites. Accessing resources that directly explain how to answer questions from the actual assignment or exam is a violation of the Honor Code.”**

<https://teachanywhere.stanford.edu/policies/academic>

- Course policy for Homeworks: **“In all cases, it is not permissible for students to enter *homework* questions into any software, apps, or websites. Accessing resources that directly explain how to answer questions from the actual assignment or exam is a violation of *course policy*.”**

Bug bounty!



- We hope all PSETs and slides will be bug-free.
- **However, we sometimes make typos.**
- **If you find a typo** (that affects understanding*) on slides, IPython notebooks, Section material or PSETs:
 - Let us know! (Post on Ed or tell a CA).
 - The first person to catch a bug gets a bonus point.



Bug Bounty Hunter

*So, typos like **thees onse** don't count, although please point those out too. Typos like **2 + 2 = 5** do count, as does pointing out that we omitted some crucial information.

For SCPD Students (and all students)

- All office hours held online (on Nooks)
- One of the recitation sections will be recorded.
- See the website for more details! (Coming soon...)

OAE forms

- Please send OAE forms to

cs161-win2021-staff@lists.stanford.edu

Feedback!

- We will have an anonymous feedback form on the course website (bottom of the main page).
- Please help us improve the course!

How are you
approaching CS 161?

Everyone can succeed in this class!

1. Work hard
2. Work smart
3. Ask for help

4. CS 161A
one unit supplementary
class (deadline to apply:
Fri, Jan 15)



The big questions

- Who are we?
 - Professor, TA's, students?
- Why are we here?
 - Why learn about algorithms?
- What is going on?
 - What is this course about?
 - Logistics?
- Can we multiply integers?
 - And can we do it quickly?

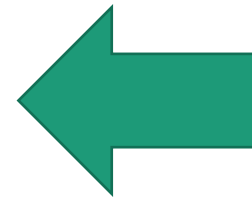


Course goals

- Think *analytically* about algorithms
- Flesh out an “*algorithmic toolkit*”
- Learn to *communicate clearly* about algorithms

Today’s goals

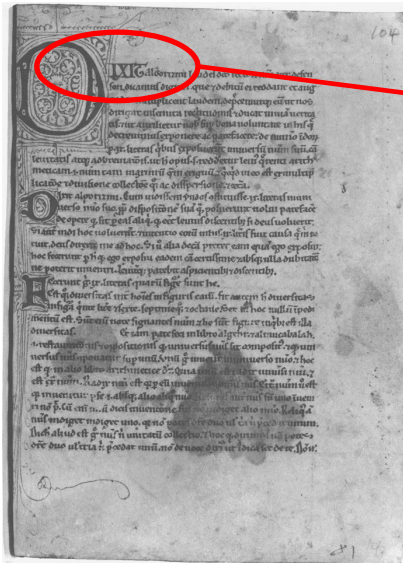
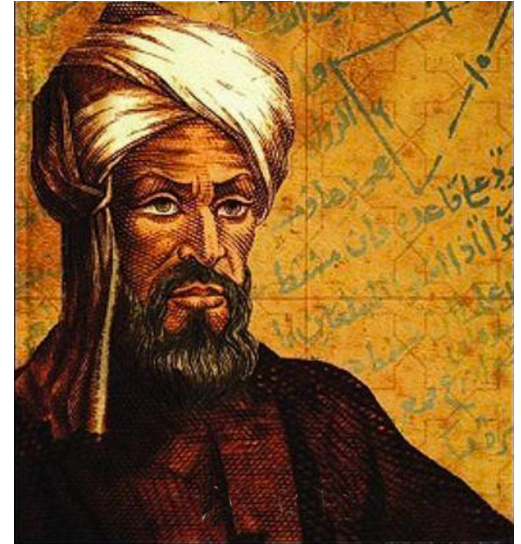
- Karatsuba Integer Multiplication
- Algorithmic Technique:
 - *Divide and conquer*
- Algorithmic Analysis tool:
 - *Intro to asymptotic analysis*



Let's start at the beginning

Etymology of “Algorithm”

- Al-Khwarizmi was a 9th-century scholar, born in present-day Uzbekistan, who studied and worked in Baghdad during the Abbasid Caliphate.
- Among many other contributions in mathematics, astronomy, and geography, he wrote a book about how to multiply with Arabic numerals.
- His ideas came to Europe in the 12th century.



Dixit algorizmi
(so says Al-Khwarizmi)

- Originally, “Algorisme” [old French] referred to just the Arabic number system, but eventually it came to mean “Algorithm” as we know today.

This was kind of a big deal

XLIV × XCVII = ?

$$\begin{array}{r} 44 \\ \times 97 \\ \hline \end{array}$$



Integer Multiplication

$$\begin{array}{r} 44 \\ \times 97 \\ \hline \end{array}$$

Integer Multiplication

$$\begin{array}{r} 1234567895931413 \\ \times 4563823520395533 \\ \hline \end{array}$$

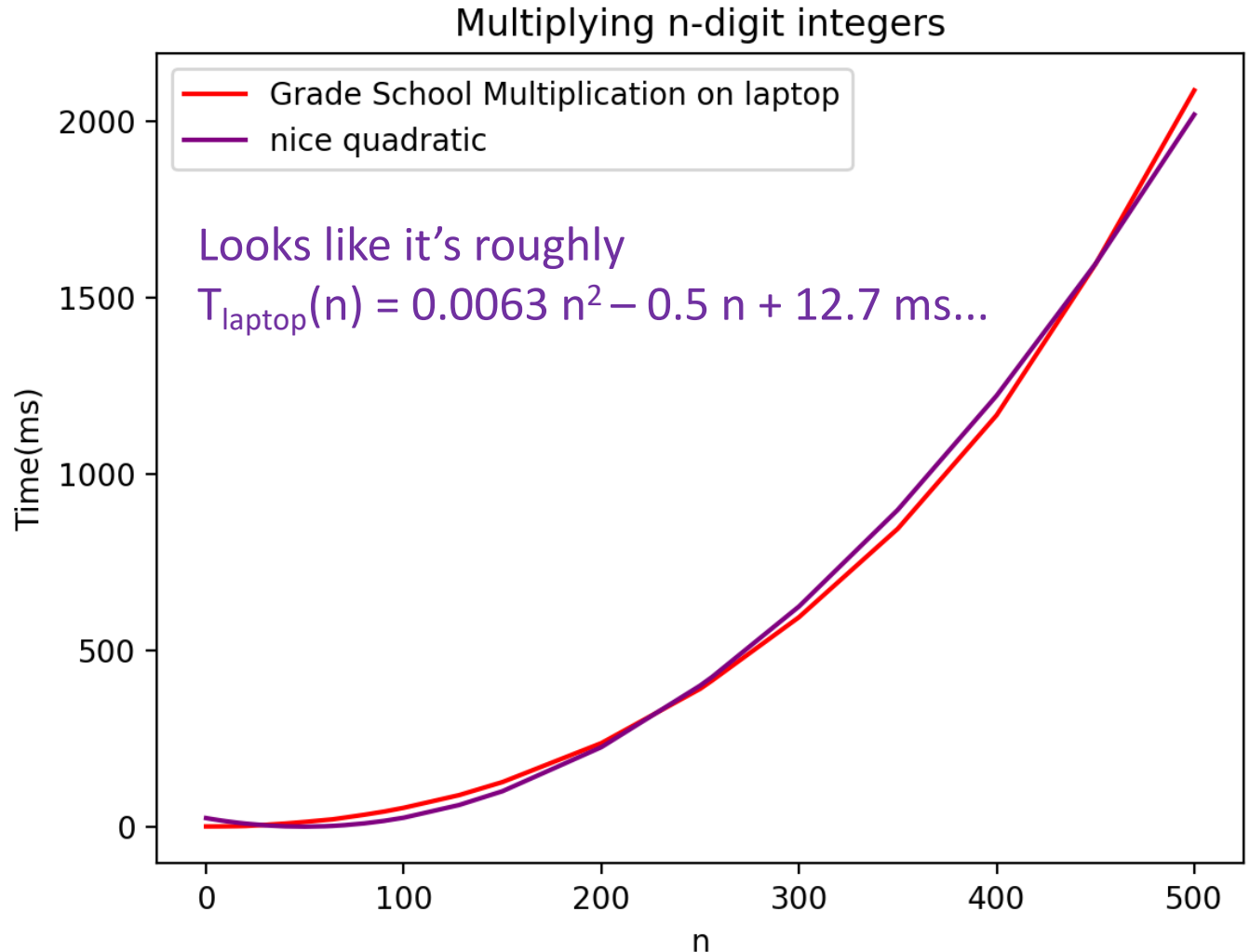
Big-Oh Notation

- We say that Grade-School Multiplication
“runs in time $O(n^2)$ ”
- Formal definition coming Wednesday!
- Informally, big-Oh notation tells us how the running time scales with the size of the input.

highly non-optimized

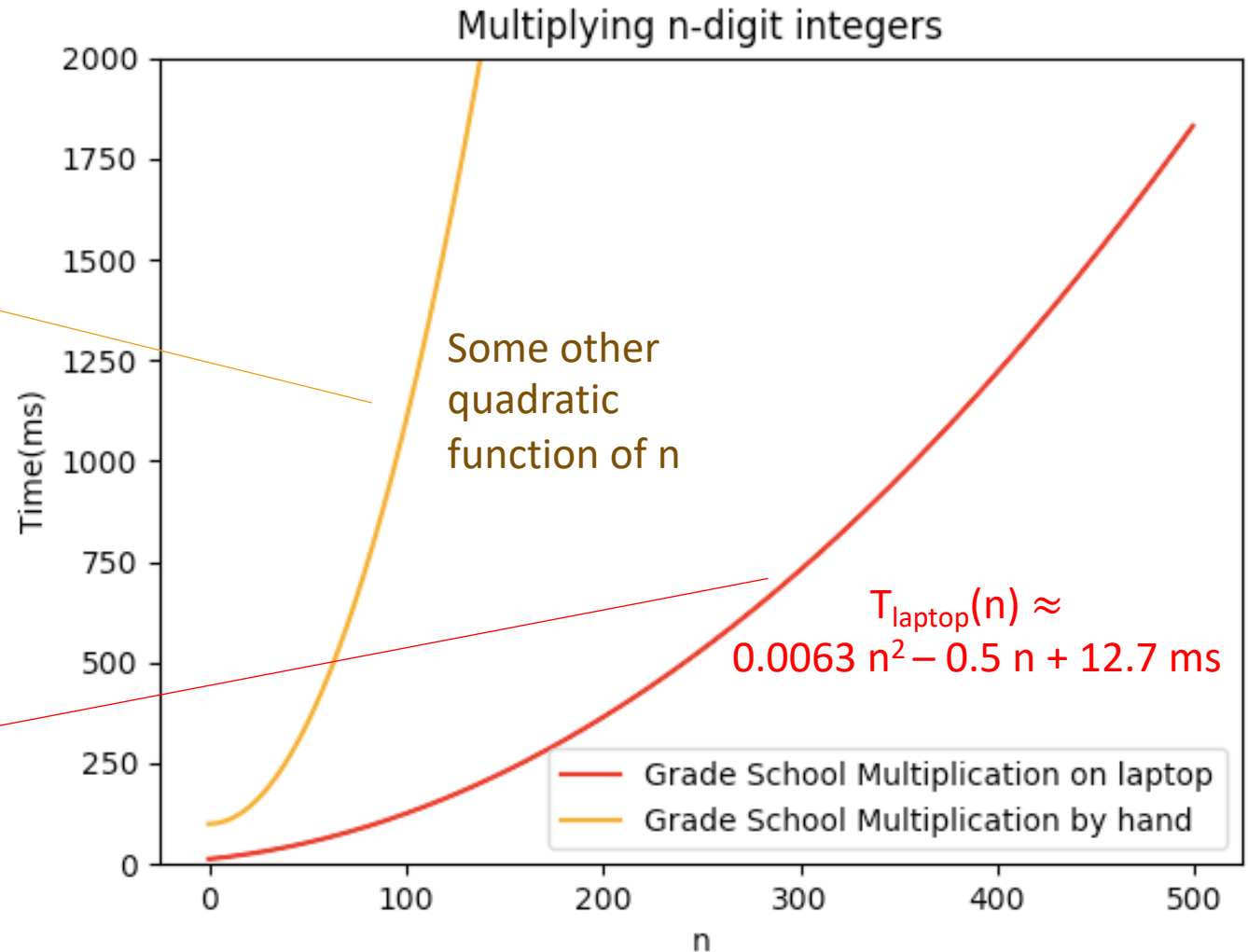
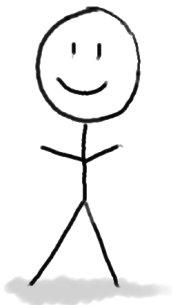
Implemented in Python, on my laptop

The runtime “scales like” n^2

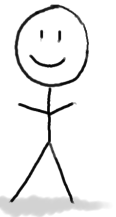
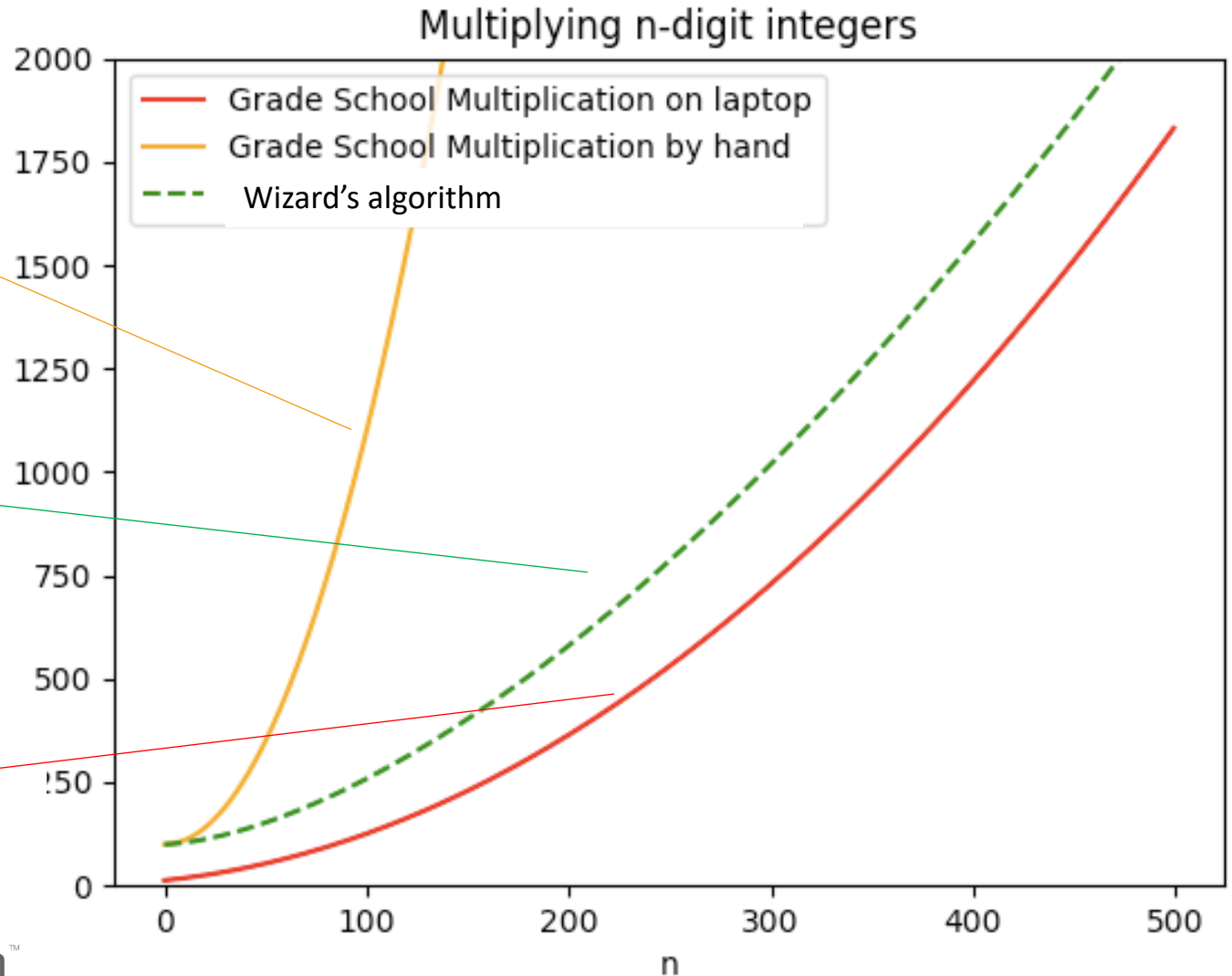


Implemented by hand

The runtime still “scales like” n^2



Why is big-Oh notation meaningful?



$$\approx \frac{n^{1.6}}{10} + 100$$




$$\approx .0063n^2$$




Let n get bigger...

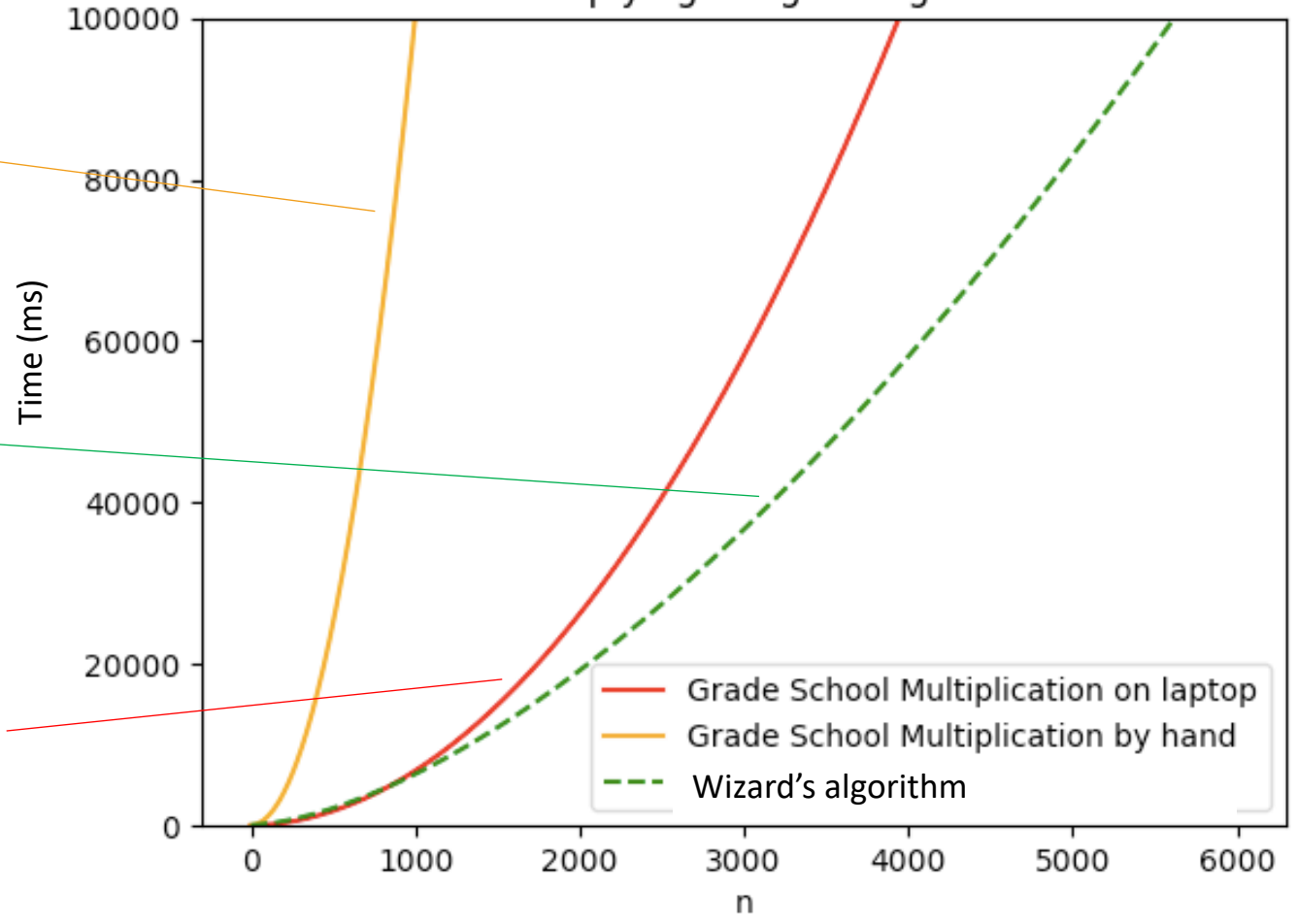
Multiplying n-digit integers



$\approx \frac{n^{1.6}}{10} + 100$



$\approx .0063n^2$

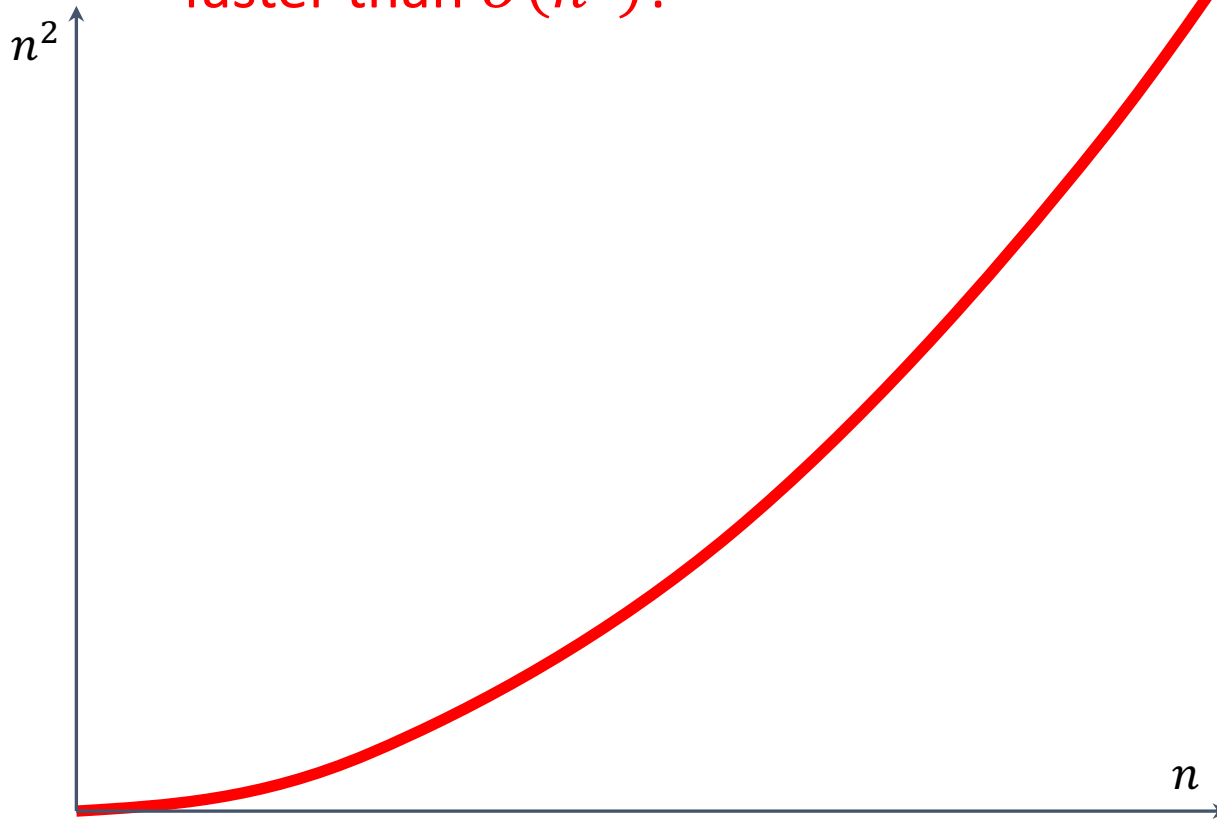


Take-away

- An algorithm that runs in time $O(n^{1.6})$ is “better” than an an algorithm that runs in time $O(n^2)$.
- So the question is...

Can we do better?

Can we multiply n -digit integers faster than $O(n^2)$?

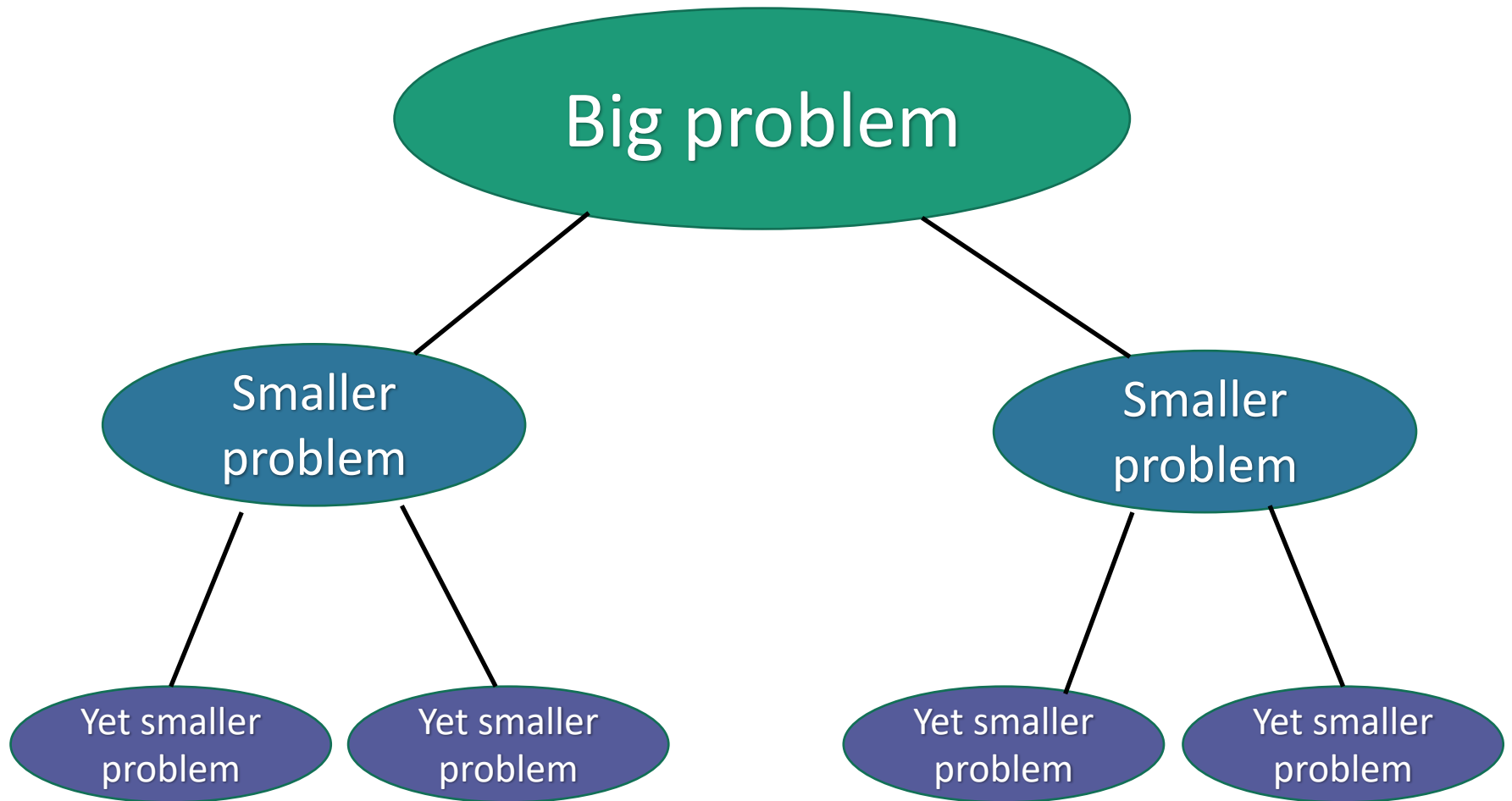


Let's dig in to our algorithmic toolkit...



Divide and conquer

Break problem up into smaller (easier) sub-problems



Divide and conquer for multiplication

Break up an integer:

$$1234 = 12 \times 100 + 34$$

$$1234 \times 5678$$

$$= (12 \times 100 + 34) (56 \times 100 + 78)$$

$$= (12 \times 56) 10000 + (34 \times 56 + 12 \times 78) 100 + (34 \times 78)$$



1



2



3



4

One 4-digit multiply



Four 2-digit multiplies

More generally

Suppose n is even



Break up an n -digit integer:

$$[x_1x_2 \cdots x_n] = [x_1x_2 \cdots x_{n/2}] \times 10^{n/2} + [x_{n/2+1}x_{n/2+2} \cdots x_n]$$

$$\begin{aligned} x \times y &= (a \times 10^{n/2} + b)(c \times 10^{n/2} + d) \\ &= \underbrace{(a \times c)}_1 10^n + \underbrace{(a \times d + c \times b)}_2 10^{n/2} + \underbrace{(b \times d)}_4 \end{aligned}$$

One n -digit multiply



Four $(n/2)$ -digit multiplies



Divide and conquer algorithm

not very precisely...

(Assume n is a power of 2...)

x, y are n -digit numbers

Multiply(x, y):

- **If** $n=1$:
 - **Return** xy

Base case: I've memorized my
1-digit multiplication tables...

- Write $x = a 10^{\frac{n}{2}} + b$

- Write $y = c 10^{\frac{n}{2}} + d$

a, b, c, d are
 $n/2$ -digit numbers

- Recursively compute ac, ad, bc, bd :

- $ac = \mathbf{Multiply}(a, c)$, etc..

- Add them up to get xy :

- $xy = ac 10^n + (ad + bc) 10^{n/2} + bd$

Make this pseudocode
more detailed! How
should we handle odd n ?
How should we implement
"multiplication by 10^n "?

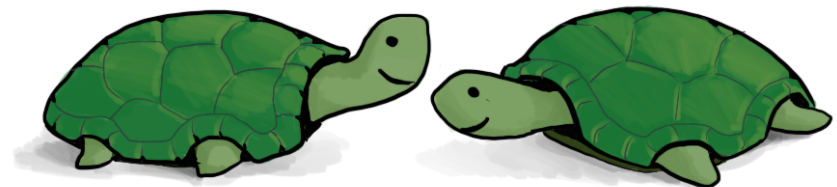


Think-Pair-Share

- We saw that this 4-digit multiplication problem broke up into four 2-digit multiplication problems

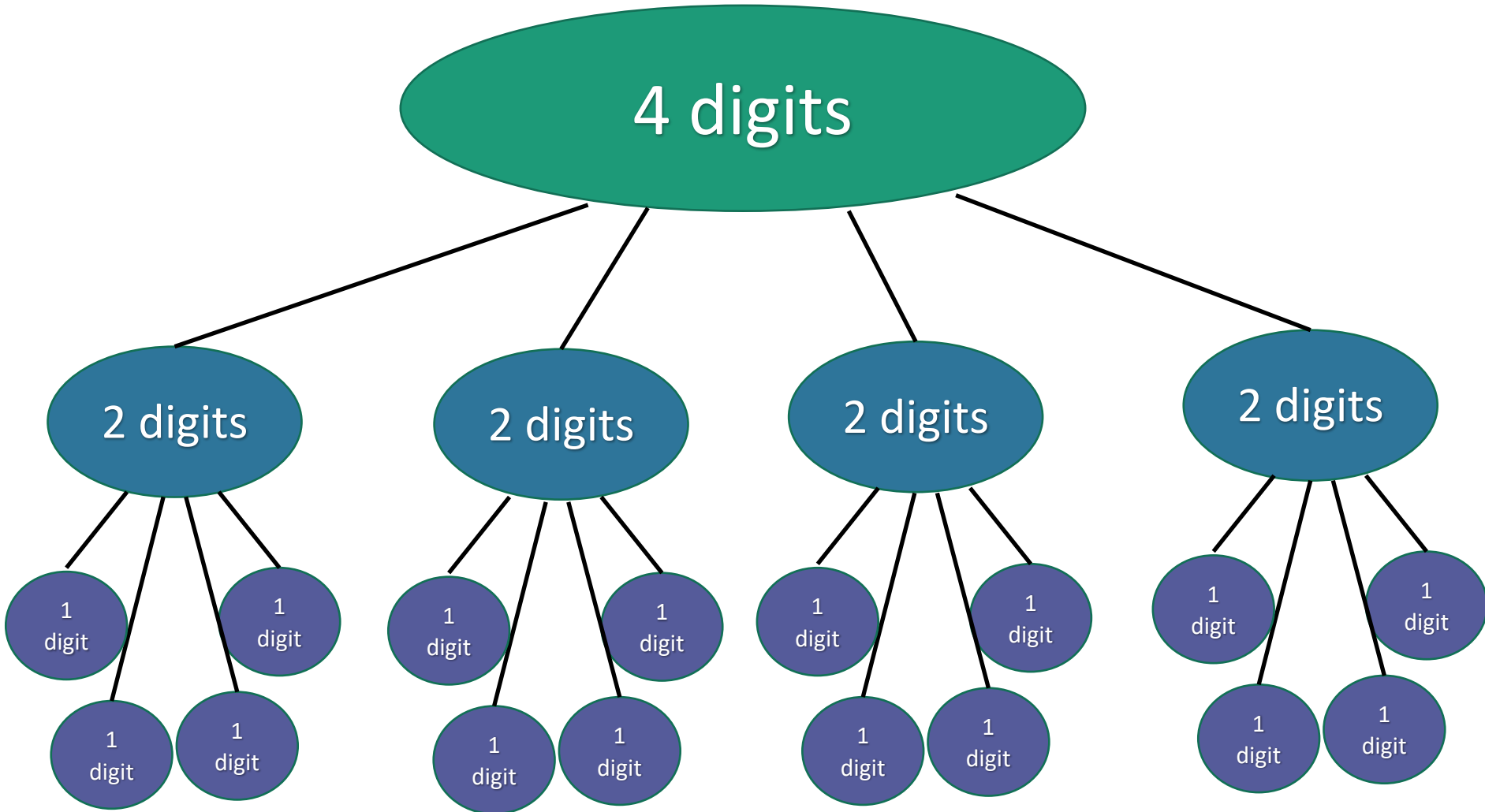
$$1234 \times 5678$$

- If you recurse on those 2-digit multiplication problems, how many 1-digit multiplications do you end up with total?



Recursion Tree

16 one-digit multiplies!



What is the running time?

- Better or worse than the grade school algorithm?
- How do we answer this question?
 1. Try it.
 2. Try to understand it analytically.

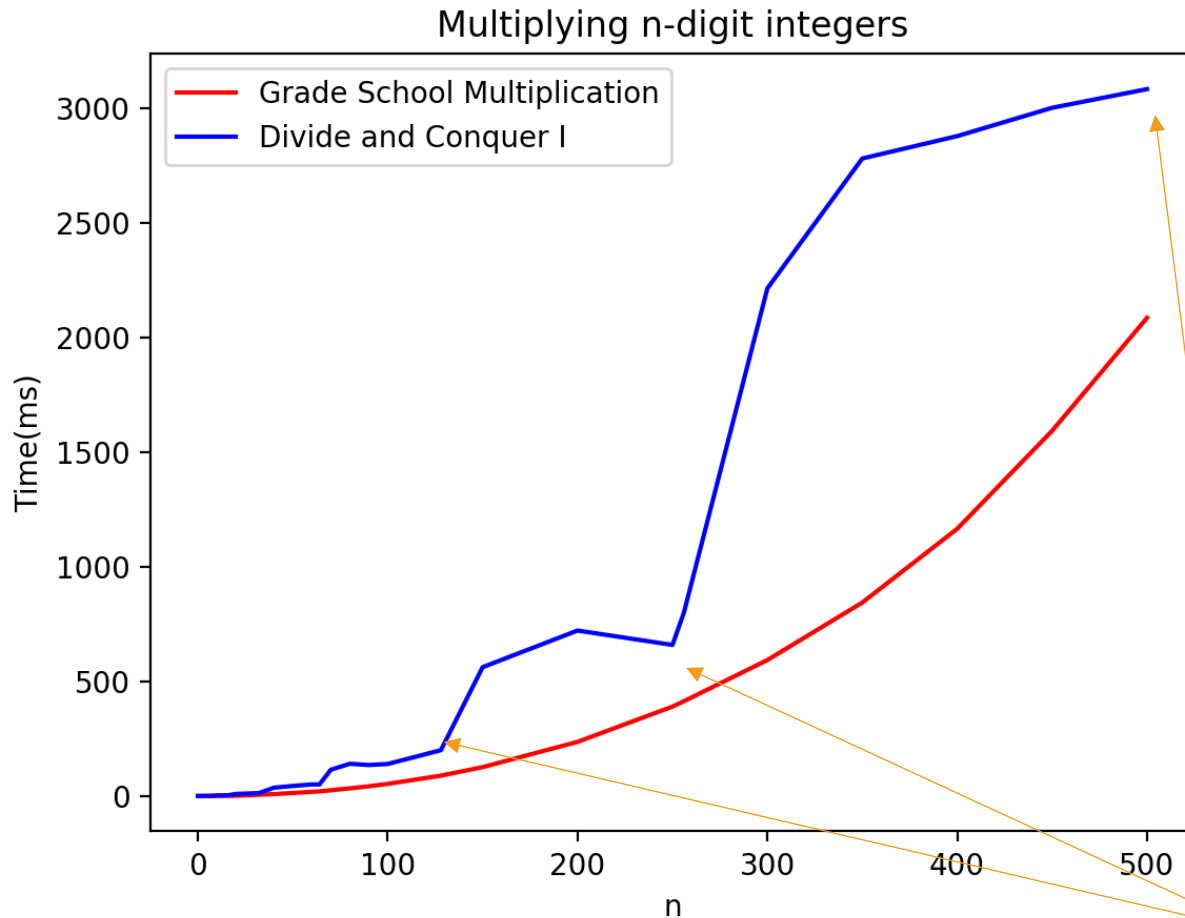
1. Try it.

Conjectures about running time?

Doesn't look too good but hard to tell...

Maybe one implementation is slicker than the other?

Maybe if we were to run it to $n=10000$, things would look different.



Something funny is happening at powers of 2...

2. Try to understand the running time analytically

- Proof by meta-reasoning:

It must be faster than the grade school algorithm, because we are learning it in an algorithms class.

Not sound logic!

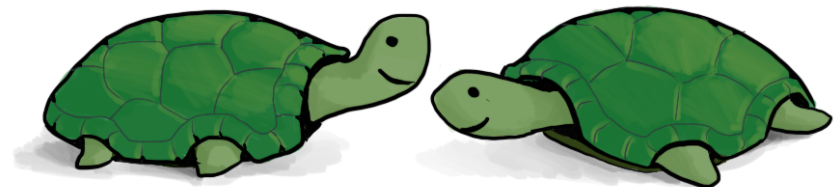


Plucky the Pedantic Penguin

2. Try to understand the running time analytically

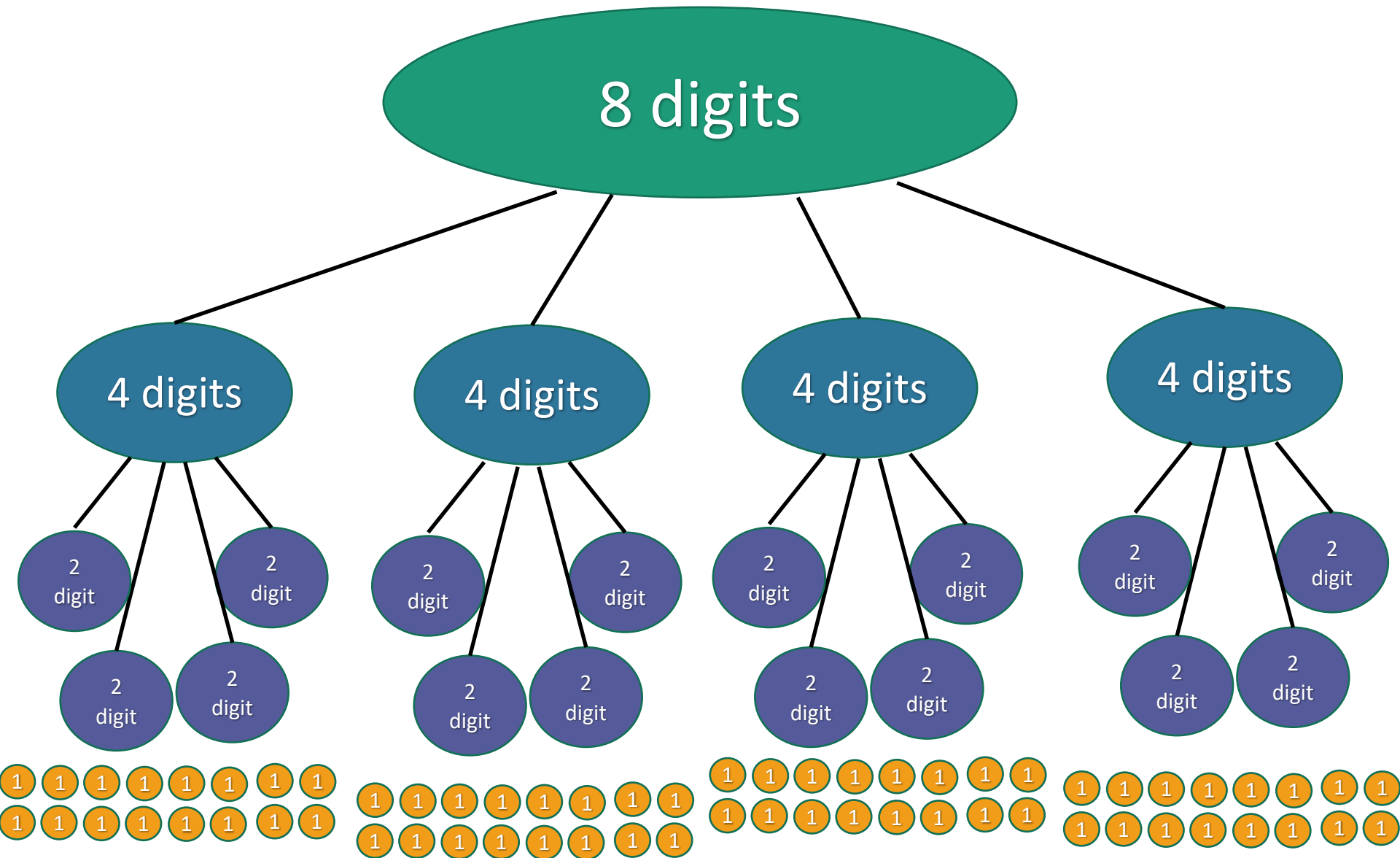
Think-Pair-Share:

- We saw that multiplying 4-digit numbers resulted in 16 one-digit multiplications.
- How about multiplying 8-digit numbers?
- What do you think about n -digit numbers?



Recursion Tree

64 one-digit multiplies!

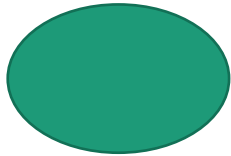


2. Try to understand the running time analytically

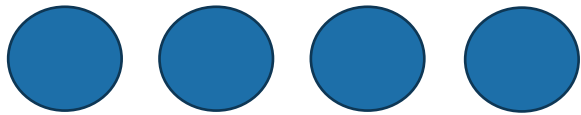
Claim:

The running time of this algorithm is
AT LEAST n^2 operations.

There are n^2 1-digit problems

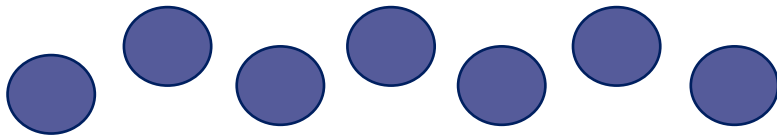


1 problem
of size n



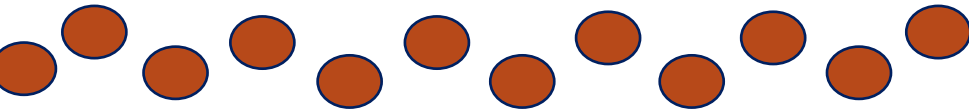
4 problems
of size $n/2$

...



4^t problems
of size $n/2^t$

...



n^2 problems
of size 1

- If you cut n in half $\log_2(n)$ times, you get down to 1.
- So at level $t = \log_2(n)$ we get...

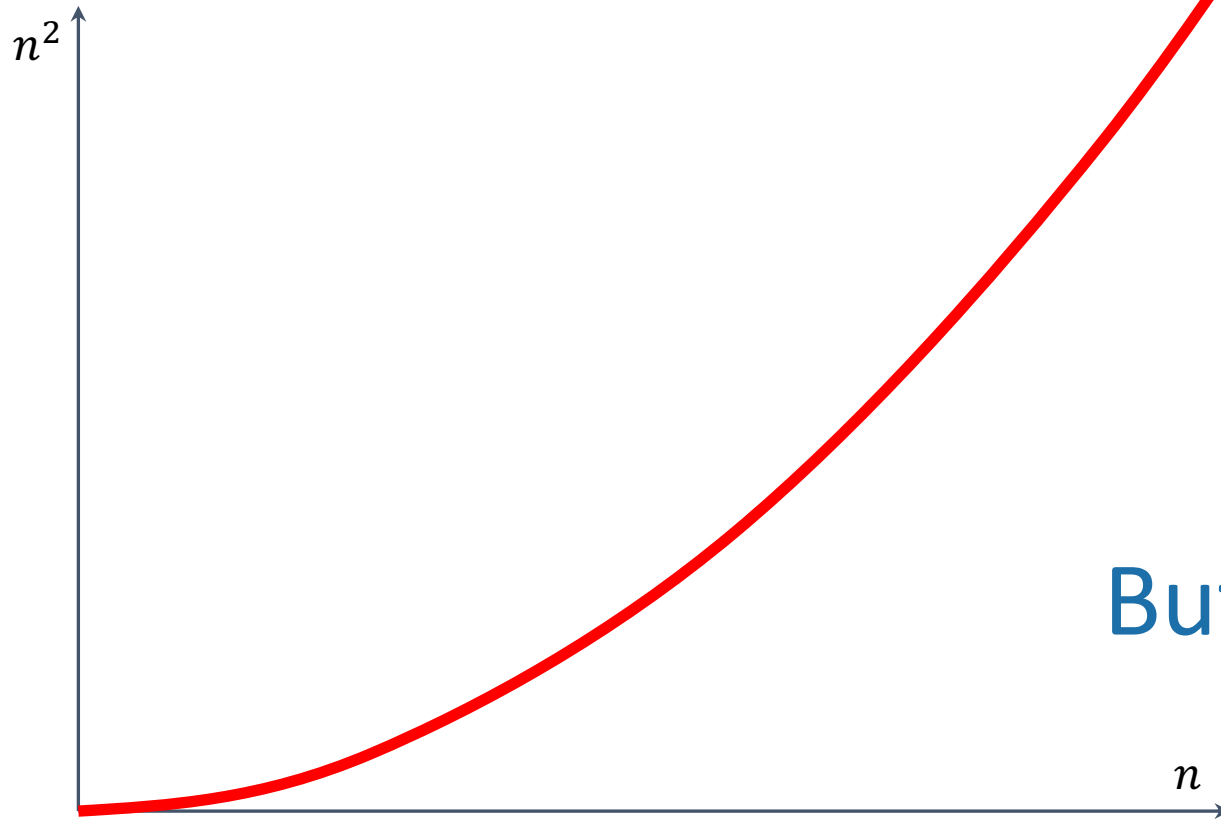
$$4^{\log_2 n} = n^{\log_2 4} = n^2$$

problems of size 1.

Note: this is just a cartoon – I'm not going to draw all 4^t circles!

That's a bit disappointing

All that work and still (at least) $O(n^2)$...



But wait!!

Divide and conquer **can** actually make progress

- Karatsuba figured out how to do this better!

$$\begin{aligned}xy &= (a \cdot 10^{n/2} + b)(c \cdot 10^{n/2} + d) \\ &= ac \cdot 10^n + (ad + bc)10^{n/2} + bd\end{aligned}$$

Need these three things



- If only we could recurse on three things instead of four...

Karatsuba integer multiplication

- Recursively compute these THREE things:

- ac
- bd
- $(a+b)(c+d)$

Subtract these off

get this

$$(a+b)(c+d) = ac + bd + bc + ad$$

- Assemble the product:

$$\begin{aligned} xy &= (a \cdot 10^{n/2} + b)(c \cdot 10^{n/2} + d) \\ &= ac \cdot 10^n + (ad + bc)10^{n/2} + bd \end{aligned}$$





How would this work?

x, y are n -digit numbers

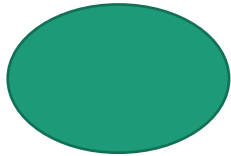
(Still not super precise, see IPython notebook for detailed code. Also, still assume n is a power of 2.)

Multiply(x, y):

- **If** $n=1$:
 - **Return** xy
- Write $x = a 10^{\frac{n}{2}} + b$ and $y = c 10^{\frac{n}{2}} + d$
- $ac =$ **Multiply**(a, c)
- $bd =$ **Multiply**(b, d)
- $z =$ **Multiply**($a+b, c+d$)
- $xy = ac 10^n + (z - ac - bd) 10^{n/2} + bd$
- **Return** xy

a, b, c, d are
 $n/2$ -digit numbers

What's the running time?

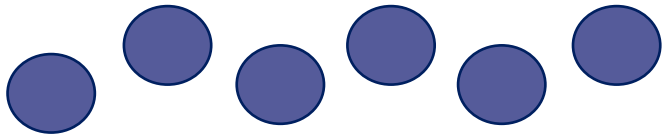


1 problem
of size n



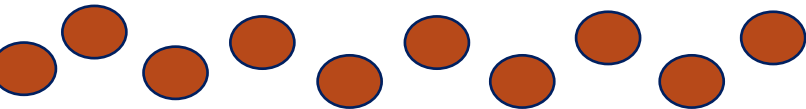
3 problems
of size $n/2$

...



3^t problems
of size $n/2^t$

...



Note: this is just a cartoon – I'm not going to draw all 3^t circles!

- If you cut n in half $\log_2(n)$ times, you get down to 1.
- So at level $t = \log_2(n)$ we get...

$$3^{\log_2 n} = n^{\log_2 3} \approx n^{1.6}$$

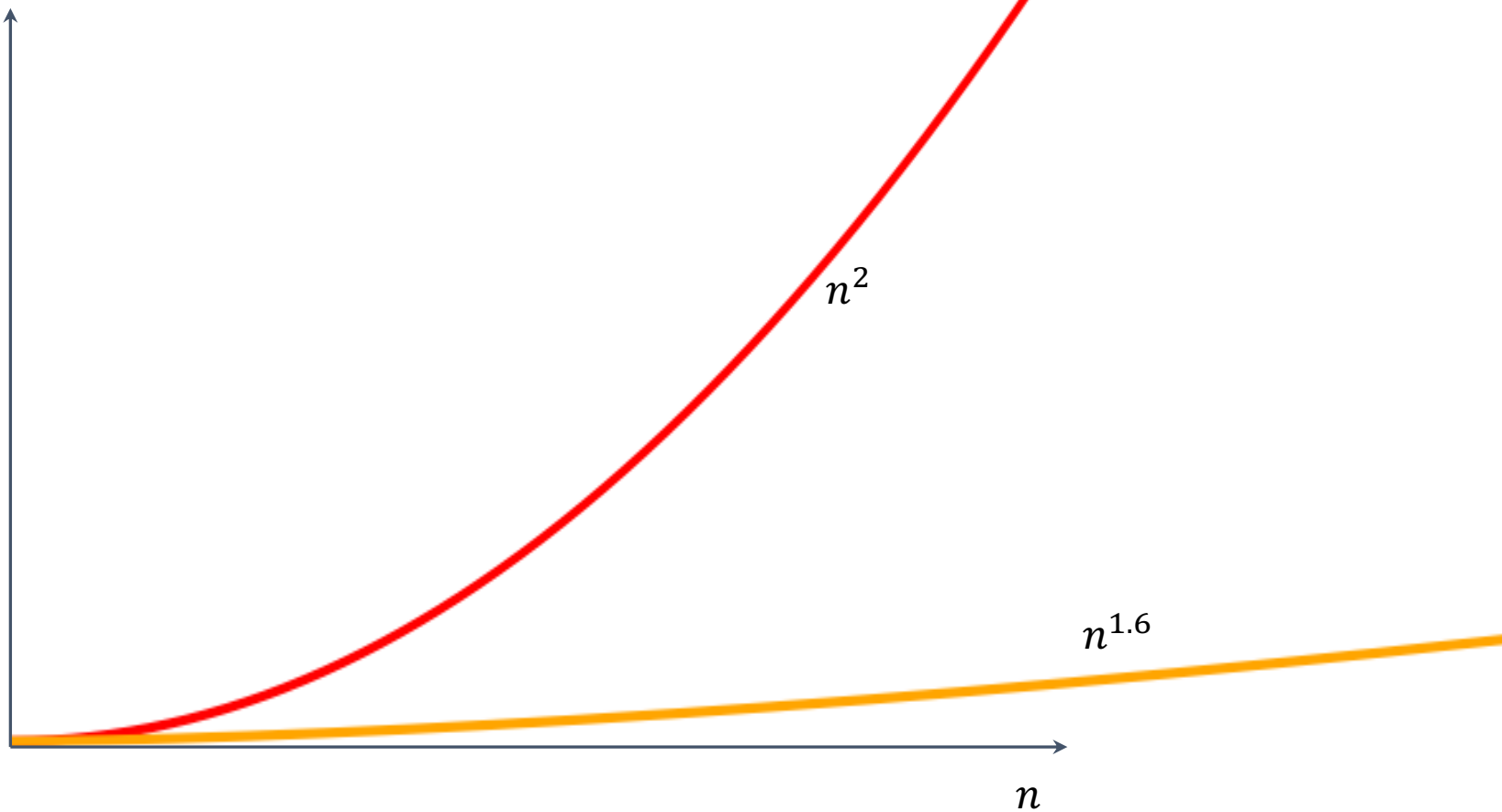
problems of size 1.

$n^{1.6}$
problems
of size 1

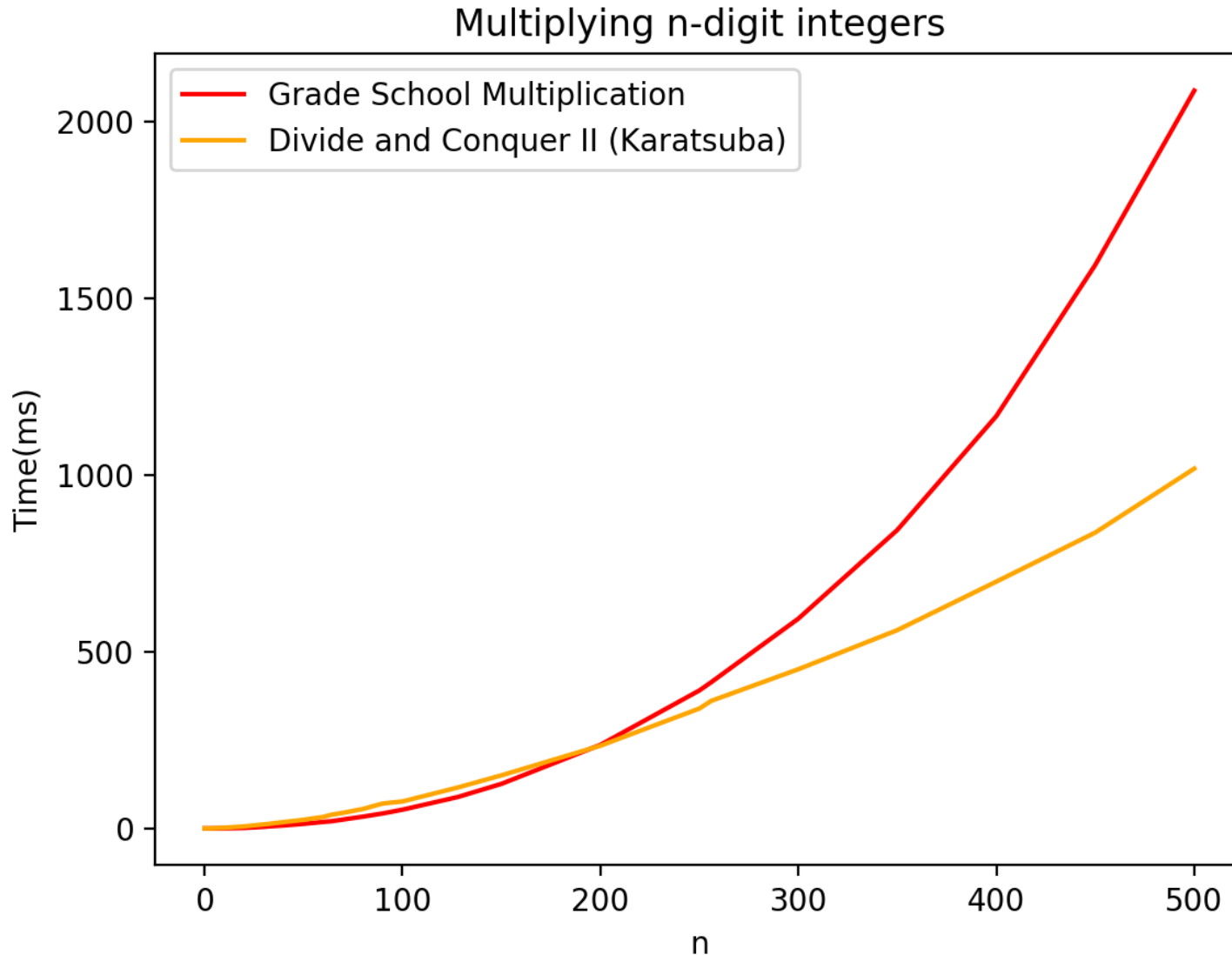
We aren't accounting for the work at the higher levels!
But we'll see later that this turns out to be okay.



This is much better!



We can even see it in real life!



Can we do better?

- **Toom-Cook** (1963): instead of breaking into three $n/2$ -sized problems, break into five $n/3$ -sized problems.
 - Runs in time $O(n^{1.465})$



Try to figure out how to break up an n -sized problem into five $n/3$ -sized problems! (**Hint: start with nine $n/3$ -sized problems**).

Given that you can break an n -sized problem into five $n/3$ -sized problems, where does the 1.465 come from?



Siggi the Studious Stork

Ollie the Over-achieving Ostrich

- **Schönhage–Strassen** (1971):
 - Runs in time $O(n \log(n) \log \log(n))$
- **Furer** (2007)
 - Runs in time $n \log(n) \cdot 2^{O(\log^*(n))}$
- **Harvey and van der Hoeven** (2019)
 - Runs in time $O(n \log(n))$

[This is just for fun, you don't need to know these algorithms!]

Course goals

- Think **analytically** about algorithms
- Flesh out an “**algorithmic toolkit**”
- Learn to **communicate clearly** about algorithms

Today's goals

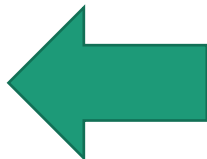
- Karatsuba Integer Multiplication
- Algorithmic Technique:
 - **Divide and conquer**
- Algorithmic Analysis tool:
 - **Intro to asymptotic analysis**




How was the pace
today?

The big questions

- Who are we?
 - Professor, TA's, students?
- Why are we here?
 - Why learn about algorithms?
- What is going on?
 - What is this course about?
 - Logistics?
- Can we multiply integers?
 - And can we do it quickly?
- Wrap-up



Wrap up

- cs161.stanford.edu
- Algorithms are fundamental, useful and fun!
- In this course, we will develop both algorithmic intuition and algorithmic technical chops 
- Karatsuba Integer Multiplication:
 - You can do better than grade school multiplication!
 - Example of divide-and-conquer in action
 - Informal demonstration of asymptotic analysis

Next time

- Sorting!
- Asymptotics and (formal) Big-Oh notation
- Divide and Conquer some more



BEFORE Next time

- ***Pre-lecture exercise!*** On the course website!
- Join Ed!